基于模拟退火算法的数据隐藏技术

摘要

近年来，随着大数据技术的不断应用，数据的保护与隐藏越来越受到人们的重视。本文主要针对不同情况下的数据隐藏问题做出了模型的建立和求解，为数据的保护提供了一定的思路和方案。

对于问题 1，我们对一般的二元数据表隐私保护问题进行了模型的建立和求解。我们引入了信息损失量、差异度距离等概念，通过建立使得信息损失量最小的最优化目标函数，来监督隐藏数据量最少。在求解过程中，我们使用了**聚类分析中的 bi-kmeans 算法**，把两条数据中的不同属性值的个数作为两条数据之间的距离，先找到簇中距离最远的两个点，最终形成新的簇。之后将隐匿信息较多的元组进行了重新聚类划分优化。最终得到附件表中二元 1 需要隐匿 20 个数据，二元 2 中需要隐匿 358 个数据。

对于问题 2，属性取值由二元变为了多元。为了提高运算准确度，我们引入了**贪心算法**来为数据初步分组并利用**模拟退火算法**来进行模型的优化。首先，我们运用贪心算法，从寻找隐匿数据量为 0 的小组开始，依次组好隐匿数目最少的数据组，以此为局部最优目标进行迭代，最终实现对数据的初步分组，获得一个较好的初始解。其次，获得初始解后，再用模拟退火算法进行优化，以隐匿数据量总数最小为目标函数，按照自行设计的新解产生方法挑选三个数据组中的一个数据，再数据组之间进行互换，不断迭代最终得到全局最优解。最终得到附件表中多元 1 需要隐匿 628 个数据，多元 2 中需要隐匿 7946 个数据。

对于问题 3，需要额外考虑个体的保护重数，即每个分组中至少需要包含p（$p \geq 2$）个元组。首先，我们优化了贪心算法思路，把p作为分组的约束条件之一，数据组内数据条个数必须大于p。其次，我们引入了的 **k-隐匿**概念。再次，我们修改了最优化目标函数的约束条件，对其进行重新求解。然后，我们继续使用第二问中的贪心算法和模拟退火算法。最终，得出表中四组数据的隐匿个数分别为，p = 2 ：“56、458、806、9547”，p = 5：“90，674，979，10373”，p = 8 ：“102，801，1063，10414”，依次为二元 1，二元 2，多元 1，多元 2。

对于问题 4，在情形 1 中，以隐藏数据量最少为目标函数，同时增加约束条件，即每个数据条的隐匿数量必须为 0，1，或全隐。为此，我们要使全隐的数据条个数尽可能少。首先，针对不同的保护重数p，使用贪心算法对原始数据集分组，具体步骤如题 3 中所示。其次，我们对模拟退火算法进行优化，当组中的数据条的隐匿数目大于 1 时，视作对所有数据隐藏数据，并以此设计代价函数。在情形 2 中，在算法分组时，我们忽略不能隐藏的数据，只关注能隐藏的数据并以此为约束条件，**优化贪心算法和模拟退火算法。**

最后，我们进行了模型的评价与改进，对以上求解过程进行了分析讨论。

[关键字] 隐私保护；聚类分析;贪心算法;模拟退火算法;k-匿名

目录

一、问题背景

1.1 问题背景

随着医疗系统、交通枢纽等流量平台信息化建设及其在商业调查和科学研究的需要，大量用户数据被收集并发布，其中包含了大量个人隐私信息，而信息共享又是不可避免的。通过适当的隐藏部分信息，就可以保证发布的数据中隐私信息不会被泄露，也能保证发布数据的有效性以供实际需要。因而，找到合适的隐藏信息的方式是亟待解决的问题。

随着大数据分析技术的迅猛发展，研究者以及各个商业公司迫切的需要从大数据中挖掘出有价值的信息。要想从大数据中挖掘信息，首先要有足够的可公开的数据，但是当大规模数据拥有者比如医院、政府、大数据公司等，对外发布数据时，不可避免的会涉及到公民的隐私问题。如果最大限度的保护公布数据的统计特征，又不泄露公民的隐私显得格外重要。

1.2 问题重述

这种通过隐藏部分数据使得每个个体落在一个无差别信息组中的隐私保护方式已经被广泛采用，而考虑到后期研究结果的反馈，隐藏数据量不宜过多，因此怎样选择数据隐藏才能使得隐私安全得到保障的同时又不至于丢失过多信息，这是当下的一个难题。在上述课题背景下，请你们小组依次考虑下面这些经过抽象后的数学问题。

（1）实际中，很多信息都可以简化为只有两个不同的选项，例如性别、近 14 天是否出省、有无家族遗传病史、是否绿码等，研究附件中的两组二元数据，保证每个个体的信息都能得到保护，分别给出一个隐藏数据量最少的方案，并建立二元数据表隐私保护的一般数学模型。

（2）考虑多元数据表，研究附件中的两组多元数据，给出一个隐藏数据量最少的方案，并建立多元数据隐私保护的一般数学模型。

（3）如果个体能够隐藏在一个至少包含p个个体的数据组中，则称该个体得到了p-重保护（$p \geq 2$），下图是一个分别有 2-重保护和 4-重保护的数据隐藏示例。建立多重保护数据隐藏的数学模型，对附件数据的p-重保护问题，分别给出$p = 3,5,8$情形的最佳隐藏方案。

（4）尝试为以下几种情形提供数据隐藏技术，尽量提供数据支撑：情形 1.多元数据表中限定了隐藏模式，例如多元 4 维数据在隐藏时只允许单隐或者全隐，即限定了(*,x,x,x),(x,*,x,x),(x,x,*,x),(x,x,x,*)以及(*,*,*,*)五种隐藏模式，其中 x 表示相同的数据；情形 2.多元高维数据中需要选出若干维度的数据全发布以确保有效性（这里假设全发布的数据不构成隐私暴露），对其余维度进行以隐私保护目的的隐藏技术，如多元 6 维数据至少 2 维全发布的示例；情形 3.用户定制的的p-重保护技术，即数据在被征集时对用户开放保护选项，每个用户根据自身隐私保护的偏好程度勾选最低保护力度（即重数），考虑所有用户定制后的隐藏方案设计。

二、问题分析

2.1 问题一的分析

题目中，将日常生活中的数据进行了简化，只考虑二元数据，因此不同元组之间的差异便可直接定义为距离，基于此距离，我们希望此值最小，即隐藏的数据量最少，以此建立单目标规划模型，使用聚类分析进行初步计算后，进行结果的优化。

2.2 问题二的分析

此问题中，将不同属性的二元数据变成了多元数据，因此处理难度上升，使用第一问中的求解方法无法解决。为了更好的进行求解，我们将第一问中差异度距离的概念进行更换，重新定义不同元组之间的距离。重新进行最优化目标方程的建立，在求解方法上，我们希望使用贪心算法将数据组按照相似度进行类的划分，但求解后获取的解只是局部最优解，需要进行继续求解。我们使用模拟退火算法，通过设置降温速度，迭代次数等参数，不断调试，期望获得最优解。

2.3 问题三的分析

第三问中要求规定数据隐藏的保护重述，即在进行分组时需要规定每组中最少的数量，我们引入的 k-隐匿的概念，k 的取值分别为$k = 3,5,8$，在不同情况下进行隐匿数据的求解。因此需要新增约束条件，进行优化模型的求解。对于求解算法，则继续使用第 2，3 问中的方法

2.4 问题四的分析

第四问选择对情形 1 和情形 2 进行求解，对于情形 1，新增约束条件，每组数据中的隐匿个数只能选择 1 或者所有，对于情形 2，规定最后两个维度的数据无法被隐藏。基于上述约束条件，分别进行模型的建立和求解，得到最终结果

三、模型假设

1. 假设不同数据之间的差异只存在于数值上，不考虑其他类型的差异
2. 认为一个数据集中所有属性的权重均相同，则不存在某个更重要的数据属性
3. 若两组数据之间不存在差异，则认为这两组数据不经过处理即可发布
4. 假设每个数据表中所有属性的维度是相同的
5. 当一个数据在隐匿之前有能得到有效保护时，不隐匿就是最优解
6. 在处理问题四的第二种情形时，默认最后两个维度的数据无法被隐匿。

四、符号说明

符号	意义
X	所有的数据
Ai	第 i 个数据
$AiBj$	第 i 个数据属性值为 j 的下标集合
Ui	等价类
Ci	等价类中的数据
Ci^*	隐匿后的数据
Ti	第 i 个等价类的数据隐匿量
T	数据隐匿总量
P	模拟退火算法中接受新解的概率
α	降温系数

五、问题一模型建立与求解

5.1 模型的准备

在建立模型之前，我们首先要了解一些数据集中常见的概念，如下：

（1）标识符：可以直接确定一个个体，如：身份证号，姓名等。

（2）准标识符集:可以和外部表链接来识别个体的最小属性集，如：邮编，生日，性别等。

（3）链式攻击：为了保护用户隐私，不让用户的信息泄露，在发布信息时，可以删去标识符信息，试图达到保护隐私的目的。但是攻击者手上如果存在其他信息表，可以通过准标识符信息进行链接，就可以推断出标识符信息。这就是链式攻击

（4）隐藏处理：为了保护用户信息，我们需要对准标识符信息进行一定的处理，通过隐藏一些数据来使表中任意数据都可以找到与其相同的数据，即对标识符进行隐藏处理。

5.2 模型的建立

在题目所给数据中，可以认为已经将标识符数据隐藏，所有数据都为准标识符数据，因此，只需要对准标识符数据进行隐藏处理[1]。

为方便讨论，假定附件数据中，共有 M 维数据；即 m 行，每个数据均有 k 个属性，即 k 列；每个属性有 n 种取值，即 n 元数据。

记所有的数据为 X，第 i 个数据为 Ai，则有：

$$X = \{A1, A2, \ldots, Am\} \quad (1)$$

记 Ai，Bj 为第 i 个数据属性值为 j 的下标集合，下标为该属性位于数据的第几列。为了使得隐藏数据量尽可能的少，现引入差异度距离和信息损失了概念。

二元信息间距离的定义：

假设两个信息间，有 n 个属性不同，则认为这两个信息的间的距离为 n，例如两个信息Am和An为间的距离记为d_{mn}。

$$d_{mn} = \sum_{j}^{k} |(A_m \mathrm{Bj} - A_n \mathrm{Bj})| \quad (2)$$

等价类的定义：

在附件数据中，属性取值相近的若干信息组成一个类，将这些信息隐匿掉部分数据，使其具有相同的属性值，便称该类为一个等价类，记为$\mathrm{Ui}(\mathrm{i} = 1,2, \ldots, \mathrm{t})$。假设一个等价类种包含 t 个数据，记为 Ci。

$$Ui = \{C1, C2, \ldots, Cn\} \quad (3)$$

等价数据的定义：

经过数据隐藏组合后的某个等价类 Ui，类中的数据经过数据隐藏之后都相同，这些信息统称为等价数据。

$$C_i^* = C_j^* (C_i^*, C_j^* \in Un)(n = 1,2, \ldots, t) \quad (4)$$

数据隐匿量的定义：

记第 i 个等价类 U 数据隐匿量为 Ti，指类中所有数据隐匿的总量，等价于数据属性不同的个数。记card（X）为 X 集合内的相同属性个数。

$$Ti = \sum_{i}^{k} card(C1Bi \cap C2Bi \ldots CtBi) \quad (5)$$

在题目所给数据中，可以认为已经将标识符数据隐藏，所有数据都为准标识符数据，因此，只需要对准标识符数据进行隐藏处理。为了使得隐藏数据量尽可能的少，现引入差异度距离和信息损失了概念。

对于表 1 与表 2 的二元组信息，数据取值只有 0，1 两种情况，可直接认为数据间的差异即为同数据的数值差异之和。

现给出元组的信息损失定义。给定元组$t, t.A_i (1 \leqslant i \leqslant d)$表示$t$在属性$A_i$上的取值,则$t$泛化为$t*$的信息损失量$IL_{\text{tuple}}$定义为:

$$IL_{\text{auple}}(t) = \sum_{i=1}^{d} w_i \cdot IL_{\text{vahue}}(t.A_i) \quad (6)$$

其中,w_i表示$t.A_i$泛化的精度损失权重.

根据定义概念，可知整个表的信息损失量为每个元组的信息损失量之和，因此给出整个表的信息损失量定义：

$$IL_{\text{iable}}(T) = \sum_{t \in T}^{T} IL_{\text{aule}}(t) \qquad (7)$$

而题目中的要求即为整个表的信息损失量值最小化，于是，我们我们可以建立单目标优化模型建立：

(一)目标函数

依据题目要求，是所有信息都得到保护的前提下，确定合理的数据隐匿方式，使得隐匿数据的总量最小，即数据损失量最小，则优化目标为：

$$\min \quad T = \sum_{i}^{m} T_i$$

(二)决策变量

考虑到隐匿量和约束条件都是关于信息属性 AiBj 的函数，因此将 AiBj 作为决策变量表示更加方便。

(三)约束条件

在式(1)、(2)、(4)的约束下，增加如下约束条件:

每个组数据的量 n 要大于 2，使得每个数据都能得到保护。

$$card(U) \geq 2 \qquad (8)$$

(2)每个数据隐匿的数据量一定大于等于不隐匿，小于等于全隐匿

$$0 \leq \sum_{j}^{n} Xij \leq m(i = 1,2, \dots, n) \qquad (9)$$

(3)分类过程需要将信息分成多个等价类，使任意两个等价类相互没有共同信息。

$$Ui \neq Uj(i \neq j)(i,j = 1,2, \dots, t) \qquad (10)$$

(四)单目标优化模型确立

目标方程：

$$\min \quad T = \sum_{i}^{m} T_i$$

约束条件：

$$\begin{cases} Ui = \{C1, C2, \dots, Cn\} \\ C_i^* = C_j^*(C_i^*, C_j^* \in Un)(n = 1,2, \dots, t) \\ Ti = \sum_{i}^{k} card(C1Bi \cap C2Bi \dots CtBi) \\ 0 \leq \sum_{j}^{n} Xij \leq m(i = 1,2, \dots, n) \\ Ui \neq Uj(i \neq j)(i,j = 1,2, \dots, t) \\ card(U) \geq 2 \end{cases}$$

5.3 模型的求解

5.3.1 初步分类

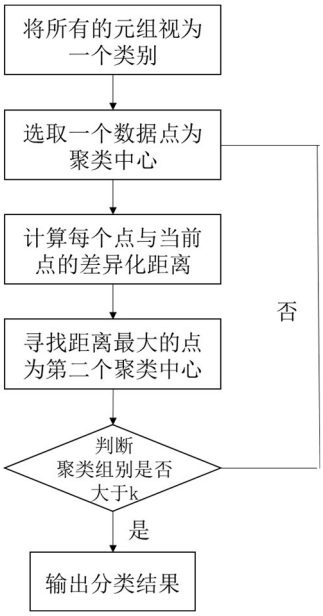
为了尽可能减小信息损失量，可以考虑根据相似度对数据进行分组，即将相似度较高的数据元组分在一组，因此可进行初步分组。[2]

对于所要求的二元数据信息损失量值最小化，我们可以使用聚类算法来进行求解。

聚类算法是按照某个特定标准(如距离)把一个数据集分割成不同的类或簇，使得同一个簇内的数据对象的相似性尽可能大，同时不在同一个簇中的数据对象的差异性也尽可能地大。也即聚类后同一类的数据尽可能聚集到一起，不同类数据尽量分离。

我们期望将所包含信息相近的元组尽可能的分在同一类中，因此考虑使用相异度距离作为聚类标准。我们根据元组间的相异度距离,将性质相异的对象形成若干类,最终形成一个聚类集.显然地,元组t被分配到类簇C中的原则是t与类簇C中质心t_{cn}的相异度距离最近.

聚类算法流程如下；



5.3.2 分类优化

如果在聚类后，存在某些类的相异度距离过大，即需要隐匿过多的数据信息，则考虑将其重新进行归类，查找是否可以将其分在其他组中，以尽可能的减小隐匿信息量[3]。

5.3.3 结果展示

第一组分类优化前隐匿数据量为 24，优化后 20

第一组分类优化前隐匿数据量为 364，优化后 346

详细数据请见附录。

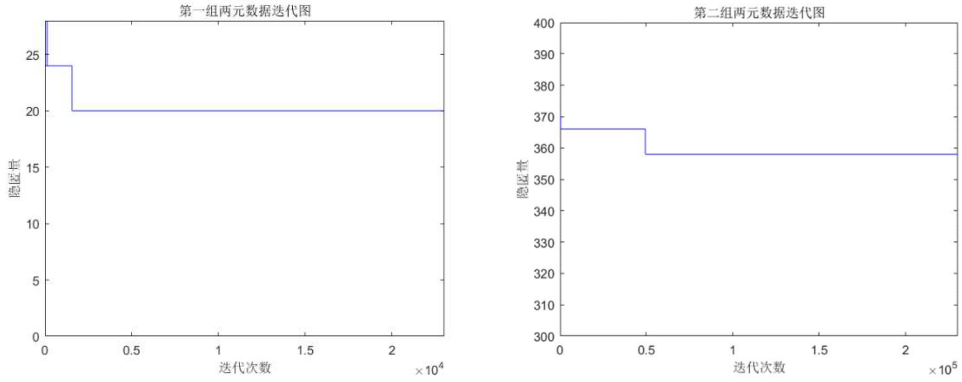


图 1 数据迭代图

六、问题二模型建立与求解

6.1 模型的建立

在问题二中，由于数据的属性取值有之前的二元，变成了多元，因此差异度距离的概念无法继续使用，为了更好的比较不同元组之间的差异度，现重新定义不同元组之间的距离。在对比两个元组时，比较其每个属性，将同一个属性出现不同的情况记录下来，计算其总和，即为元组之间的距离[4]：

给定元组t_i和t_j,两者之间的距离 Distance(t_i, t_j)定义为:

$$\text{Distance}(t_i, t_j) = \sum_{k=1}^{d} ||t_i, A_k| - |t_j. A_k|| \qquad (11)$$

其中,$||t_i, A_k| - |t_j, A_k||$表示元组$t_i$和$t_j$在属性$A_k$上的距离,对于连续型禹性,采用两者的区间长度之差,对于离散型属性,采用两者值域的基数之差.

因此将问题一的单目标最优化模型进行修改，得到新的最优化模型

目标方程：

$$\text{Distance}(t_i, t_j) = \sum_{j}^{k} |(A_m\text{Bj} - A_n\text{Bj})|$$

约束条件：

$$\begin{cases} Ui = \{C1, C2, \dots, Cn\} \\ C_i^* = C_j^*(C_i^*, C_j^* \in Un)(n = 1,2, \dots, t) \\ Ti = \sum_i^k card(C1Bi \cap C2Bi \dots CtBi) \\ 0 \le \sum_j^n Xij \le m(i = 1,2, \dots, n) \\ Ui \ne Uj(i \ne j)(i,j = 1,2, \dots, t) \\ card(U) \ge 2 \end{cases}$$

6.3 模型求解

6.3.1 基于贪心算法对数据进行初步分组

与第二问相同，我们任然需要进行初始分组，由于多元数据的属性取值变为多元，并且数据量大幅增加，而原先的算法在复杂度和运算速度上已经不适合在进行第二问的计算，因此考虑寻找新的算法进行求解。

在进行初步分类是，使用贪心算法进行求解，它是寻找最优解问题的常用方法，这种方法模式一般将求解过程分成若干个步骤，但每个步骤都应用贪心原则，选取当前状态下最好/最优的选择（局部最有利的选择），并以此希望最后堆叠出的结果也是最好/最优的解。[5]

按照贪心算法的思想，按一定的规则进行遍历，步骤如下：

Step1：从第一个数开始遍历，首先完全相同的数据分为一类，如果没有，则先不分组。

Step2：再从未分组的数据开始遍历，将最先匹配到的差异只有一个属性的数据与其分为一类，如果没有，则先不分组。

Step3：与上一部相似，开始匹配差异更多属性的数据。

Step4：重复步骤 3，直至所有数据都完成分组。

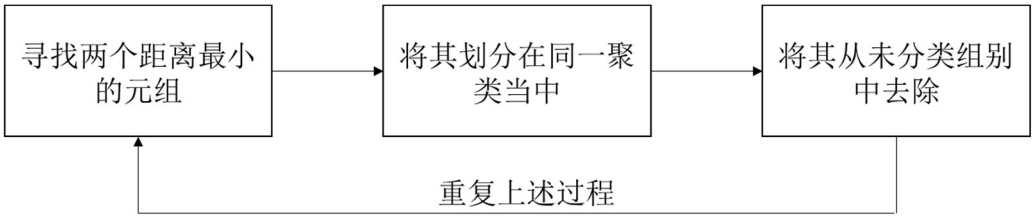


图 2 贪心算法流程图

6.3.2 利用退火算法对分组优化

使用贪心算法进行求解后，只能得到局部最优解，而第一问中的分类优化算法计算时间长，次数多，难以进行大量，多元数据的求解，因此采用模拟退火算法进行求解.

模拟退火算法最早的思想是由 N.Metropolis 等人于 1953 年提出[6]。它是基于迭代求解策略的一种随机寻优算法，其出发点是基于物理中固体物质的退火过程

与一般组合优化问题之间的相似性。该算法的优点为求解速度快，可达到全局最优解。

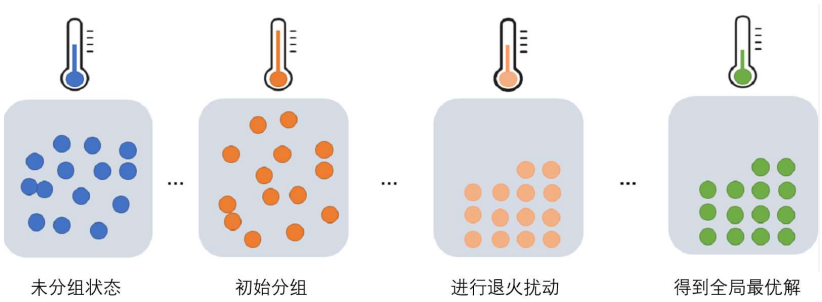


图 3 退火算法示意图

模拟退火算法包括三函数两准则，即状态产生函数，状态接受函数，温度更新函数；内循环终止和外循环终止准则[6]。这些环节之间影响模拟退火算法的优化性能。

求解模拟退火算法描述如下：

(1)解空间。解空间 S 可表示为数据的所有分组形式的排列组合。

(2)目标函数。目标函数为数据隐匿总量。要求

$$\min \quad T = \sum_{i}^{m} T_i$$

(3)新解的产生。利用上一步迭代分好的组，每次选取三个组，每组内各挑选一个数据进行交换，得到新的分组。

选组要求为：1.每次选取三个不同的组；2.三个小组两两之间隐匿的个数不能完全相同。

(4)代价函数差。假设原先的分组数据隐匿量为 T1，新的分组数据隐匿量为 T2。隐匿量差可表示为：

$$\Delta T = T2 - T1 (12)$$

(5)接受准则

$$P = \begin{cases} 1, \Delta T < 0 \\ exp(-\Delta T/T), \Delta T \geq 0 \end{cases} \qquad (13)$$

如果$\Delta T < 0$，则接受新的分组；否则，以概率$exp(-\Delta T/T)$接受新的分组，即用计算机产生一个[0,1]区间上均匀分布的随机数 rand，若rand <= $exp(-\Delta T/T)$则接受。

(6)降温。利用选定的降温系数 α 进行降温，取新的温度 T 为 AT(这里 T 为上一步迭代的温度)，这里选 α=0.999。

(7)结束条件。用选定的终止温度$e = 10^{-30}$，判断退火过程是否结束。若 T<e，则算发结束，输出当前状态。

其算法的步骤具体为：

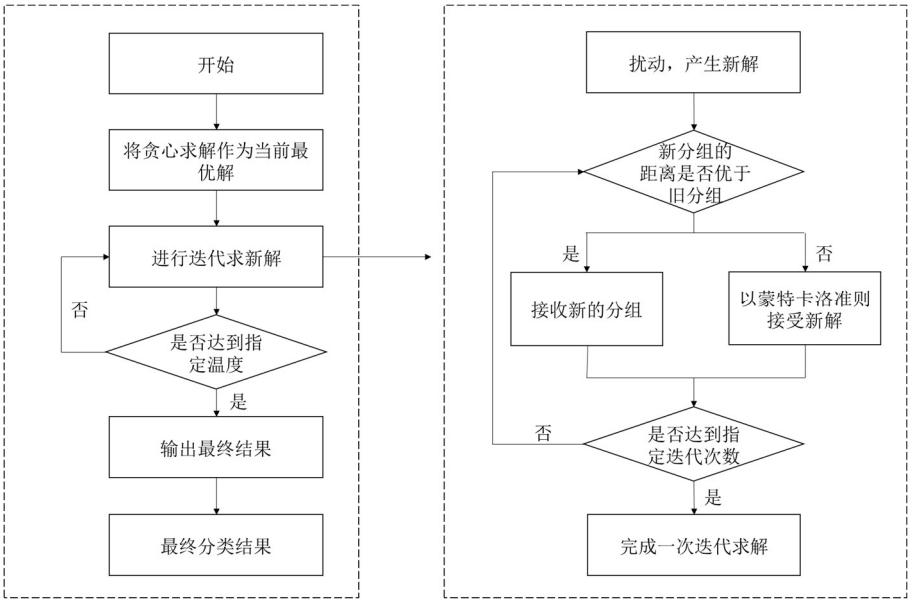


图 3 模拟退火算法流程图

6.4 结果展示

第一组分类优化前隐匿数据量为 640，优化后 628

第一组分类优化前隐匿数据量为 7964，优化后 7946

详细结果间附录。

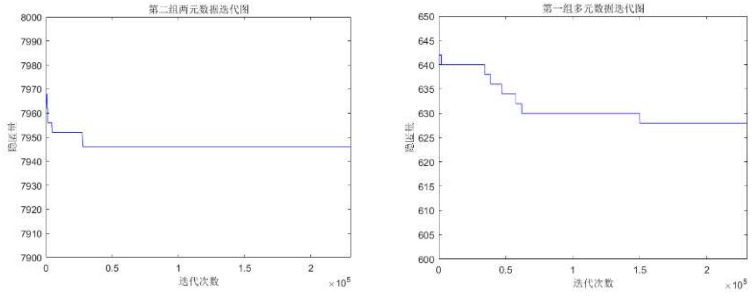


图 4 模拟退火算法迭代图

七、问题三模型建立与求解

7.1 模型准备

7.1.1 k-匿名技术

通过 k-匿名技术，可以使攻击者无法知道某特定个人是否在公开的数据中；给定一个人，攻击者无法确认他是否有某项敏感属性[7]；攻击者无法确认某条数

据对应的是哪个人。这就要求使每个个体能够隐藏在一个至少包含 k 个个体的数据组中，这样的问题成为 k 匿名问题，就等价于 P 重保护问题。

将附件数据分成 P 个等价类，每个等价类 C 至少包含 P 个等价元组，附件数据满足 k-匿名，此时所有数据都得到了 P 重保护。

在对于匿名组的划分上，匿名化数据集中包含的匿名组越多，匿名组的规模约小，数据集的可用性就越高。因而，在 P 重保护时，最好的结果是将信息分为 n/P 组。

7.2 模型建立

(一)目标函数

依据题目要求，是所有信息都得到保护的前提下，确定合理的数据隐匿方式，使得隐匿数据的总量最小，即数据损失量最小，则优化目标为：

$$\min \quad T = \sum_{i}^{m} T_i$$

(二)决策变量

考虑到隐匿量和约束条件都是关于信息属性 AiBj 的函数，因此将 AiBj 作为决策变量表示更加方便。

(三)约束条件

在式(1)、(2)、(4)、(8)、(10)的约束下，增加约束条件:

（1）每个组数据的量 n 要大于 P，使得每个数据都能得到 P 重保护。

$$card(U) \geq P(14)$$

(四)单目标优化模型确立

$$\begin{cases} Ui = \{C1, C2, \dots, Cn\} \\ C_i^* = C_j^*(C_i^*, C_j^* \in Un)(n = 1,2, \dots, t) \\ Ti = \sum_i^k card(C1Bi \cap C2Bi \dots CtBi) \\ 0 \leq \sum_j^n Xij \leq m(i = 1,2, \dots, n) \\ Ui \neq Uj(i \neq j)(i,j = 1,2, \dots, t) \\ card(U) \geq P \end{cases}$$

7.3 模型求解

7.3.1 基于贪心算法对数据进行分组

与第二问模型求解类似，我们仍然用贪心算法进行分组，但是由于要做到 P 重保护，即每个组的数量至少要为 P 个，所以要对贪心策略进行改进。

按照贪心算法的思想，按改进的规则进行遍历，步骤如下：

Step1：从第一个数开始遍历，首先完全相同的数据分为一类，如果没有，则先不

分组。

Step2：从分好的第一个类开始遍历，在未分类的数据中，寻找与类差异最小的数据放入类中的，直至类中的数据到达 P 个。

Step3：如果剩余未分组数据大于 P 个，再从未分组的数据开始遍历，将最先匹配到的差异只有一个属性的数据与其分为一类，如果没有，则先不分组；如果剩余数据未分组数据小于 P 个，寻找与其最相近的类并加入，并结束算法。

Step4：重复步骤 2。

Step5：与上一部相似，开始匹配差异更多属性的数据。

Step6：重复步骤 3~5，直至所有数据都完成分组。

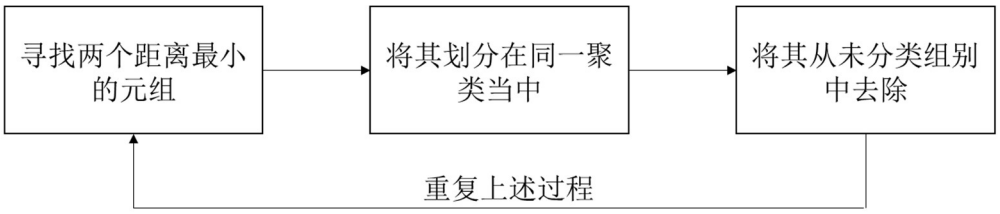


图 5 分组流程图

得到对数据的初步分组，并计算出此时的数据隐匿量

7.3.2 利用退火算法对分组优化

模拟退火算法包括三函数两准则，即状态产生函数，状态接受函数，温度更新函数[8]；内循环终止和外循环终止准则。这些环节之间影响模拟退火算法的优化性能。为了能使算法求解结果尽量靠近全局最优解，模拟退火算法可以接受较差解，当当前迭代产生的解比目前最优解更优时，算法以概率 1 接受该解作为新的最优解；而当前迭代产生 18 的解比目前最优解差时，算法则会 exp(−ΔE/T)的概率接受该差解作为新的最优解其中 T 为当前温度，ΔT 为当前解所对应的目标函数值与最优解所对应的目标函数值的差值。可以看出，当初始温度足够高，降温足够慢时，每一温度下抽样足够长、最终温度趋近于 0 时，最终降低概率 1 收敛到全局最优解。

7.4 结果展示

部分数据展示

隐匿个数	组数
0	39
1	26
2	15

表 1 二元组 1 的数据 3 重保护

隐匿个数	组数
0	6
1	43
2	69
3	60
4	13
5	9

表 2 二元组 2 的数据 3 重保护

隐匿个数	组数
2	18
3	49
4	76
5	40
6	4
7	9
8	4

表 3 多元组 1 的数据 3 重保护

隐匿个数	组数
8	3
9	22
10	71
11	166

表 4 多元组 2 的数据 3 重保护

其余结果见附录。

八、问题四模型建立与求解

8.1 情形一模型

8.1.1 目标优化模型建立

(一)目标函数

依据题目要求，是所有信息都得到保护的前提下，确定合理的数据隐匿方式，使得隐匿数据的总量最小，即数据损失量最小，则优化目标为：

$$\min \quad T = \sum_{i}^{m} T_i$$

(二)决策变量

考虑到隐匿量和约束条件都是关于信息属性 AiBj 的函数，因此将 AiBj 作为决策变量表示更加方便。

(三)约束条件

在式(1)、(2)、(4)、(8)、(10）的约束下，增加约束条件:

（1）每个数据隐匿的数量只能为 0，1 或者全隐匿

$$\sum_{j}^{n} Aij = 0，\ 1，\ k, (i = 1,2, \dots, n)(15)$$

(四)单目标优化模型确立

$$\begin{cases} Ui = \{C1, C2, \dots, Cn\} \\ C_i^* = C_j^*(C_i^*, C_j^* \in Un)(n = 1,2, \dots, t) \\ Ti = \sum_i^k card(C1Bi \cap C2Bi \dots CtBi) \\ \sum_j^n Aij = 0，\ 1，\ k, m(i = 1,2, \dots, n) \\ Ui \neq Uj(i \neq j)(i, j = 1,2, \dots, t) \\ card(U) \geq P \end{cases}$$

8.1.2 模型结果展示

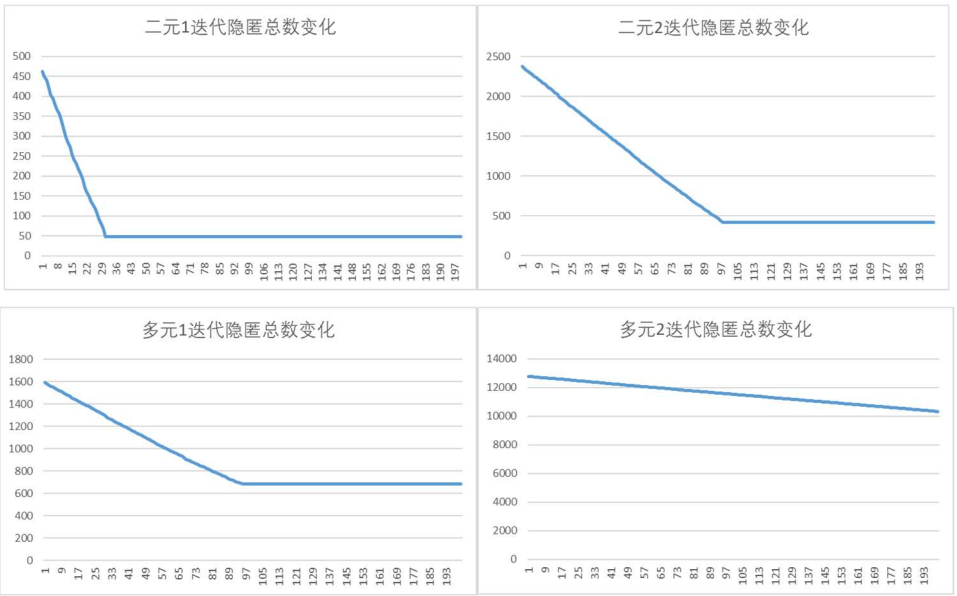


表 6 算法迭代示意图

8.1 情形二模型

8.2.1 单目标优化模型建立

我们假设数据 2 维全发布，取最后两个属性不发生改变

目标假设与决策变量与情形 1 相同，而约束条件发生改变。

(三)约束条件

在式(1)、(2)、(8)、(10）的约束下，增加约束条件:

（1）由于每个数据**最后两个属性要全发布，所以隐匿量要小于 k-2**

$$0 \le \sum_{j}^{n} Aij \le k-2(i=1,2,\dots,n) \quad (16)$$

（2）隐匿后的数据

$$C_i^*\mathrm{Bj} = C_i\mathrm{Bj}(\mathrm{j}=\mathrm{k},\mathrm{k}-1) \quad (17)$$

(四)单目标优化模型确立

$$\begin{cases} C_i^*\mathrm{Bj} = C_i\mathrm{Bj}(\mathrm{j}=\mathrm{k},\mathrm{k}-1) \\ C_i^* = C_j^*\left(C_i^*, C_j^* \in Un\right)(n=1,2,\dots,t) \\ Ti = \sum_i^k card(C1Bi \cap C2Bi \dots CtBi) \\ 0 \le \sum_j^n Aij \le k-2(i=1,2,\dots,n) \\ Ui \ne Uj(i \ne j)(i,j=1,2,\dots,t) \\ card(U) \ge P \end{cases}$$

8.2.2 结果展示

详细结构见附录

九、模型的评价

本文主要提到了贪心算法、聚类算法和模拟退火算法。通过贪心算法或聚类算法进行分组，得到初始较优解，在通过模拟退火算法进行优化。对比贪心算法和聚类算法，从算法速度看，当处理二元数据时，贪心算法时间复杂度为 n2，处理多元数据时，时间复杂度为 n3；而聚类算法在处理二元数据或多源数据时，时间复杂度均为 n2。在实践中，处理多元数据，聚类算法确实比贪心算法快。从算法效率看，贪心算法比聚类算法优异许多，聚类算法难以对噪音点进行有效的分组，而贪心算法一直在以局部最优为目标进行分组。最后用模拟退火算法优化初始较优解，我们采用自行设计的三组数据互换方法产生新解，该算法优化效果明显。

8.1 模型优点

1.在对数据的分组隐匿时，采用贪心算法和聚类算法进行分组，又使用模拟退火算法进行分组优化，能使得隐匿数量较小，使模型结果较好。

2.模型的约束条理清晰，通过多重线性进行约束，求解变得高效，且易于实现。

8.2 模型的缺点

1.在使用贪心算法对多元数组进行多重保护时，时间复杂度较高，程序运行时间较慢。

2.由于比赛时间限制，未能对模型进行进一步的优化，第四问只分析了前两问。

参考文献

[1]LIUFei,刘斐,FANHua,等.一种新型 k 匿名隐私保护算法[J].中国计算机学会,2012.

[2]于娟,韩建民,郭腾芳,等.基于聚类的高效 k-匿名化算法[J].计算机研究与发展,2009,46(s2):000480-486.

[3]陈华根,吴健生,王家林,等.模拟退火算法机理研究[J].同济大学学报：自然科学版,2004,32(6):4.

[4]孙吉贵,刘杰,赵连宇.聚类算法研究[J].软件学报,2008(1):14.

[5]张敏,于剑.基于划分的模糊聚类算法[J].软件学报,2004,15(6):11.

[6]岳琪,沈冰.模拟退火算法在单目标规划问题中的应用[J].信息技术,2006,30(5):3.

[7]LiJ,WongCW,FuWC,etal.Achievingk-AnonymitybyClusteringinAttributeHierarchicalStructures[C]//InternationalConferenceonDataWarehousing&KnowledgeDiscovery.SpringerBerlinHeidelberg,2006.

[8]张岐山,郭昆.一种基于聚类的数据流隐私保护算法[J].福州大学学报：自然科学版,2014(1):10.

附录

附录一

分组序号	元组数量	隐匿数量	分组序号	元组数量	隐匿数量
1	3	0	19	2	0
2	2	0	20	2	0
3	3	0	21	3	0
4	3	0	22	2	0
5	2	0	23	2	1
6	2	0	24	2	2
7	2	0	25	2	0
8	3	0	26	2	0
9	4	0	27	2	0
10	2	0	28	2	0
11	2	0	29	2	0
12	3	0	30	2	1
13	2	0	31	2	1
14	2	0	32	2	1
15	2	0	33	2	1
16	2	0	34	2	1
17	2	0	35	2	1
18	2	0	36	2	1

表：问题 1 第一组数据

分组序号	元组数量	隐匿数量	分组序号	元组数量	隐匿数量
1	2	0	51	2	2
2	2	0	52	2	2
3	2	0	53	2	2
4	2	0	54	2	2
5	2	0	55	2	2
6	2	2	56	2	3
7	2	1	57	2	2
8	2	1	58	2	3
9	2	1	59	2	2
10	2	3	60	2	2
11	2	1	61	2	2
12	2	1	62	2	2
13	2	1	63	2	2
14	2	1	64	2	2
15	2	2	65	2	2
16	2	1	66	2	2
17	2	1	67	2	2
18	2	1	68	2	2
19	2	1	69	2	1
20	2	1	70	2	2
21	2	1	71	2	2
22	2	2	72	2	1
23	2	1	73	2	2
24	2	1	74	2	2
25	2	1	75	2	3
26	2	3	76	2	2
27	2	1	77	2	2
28	2	1	78	2	2
29	2	1	79	2	3
30	2	1	80	2	2
31	2	1	81	2	2
32	2	1	82	2	2
33	2	1	83	2	2
34	2	1	84	2	3
35	2	1	85	2	3
36	2	2	86	2	2
37	2	2	87	2	2
38	2	2	88	2	3
39	2	2	89	2	3
40	2	2	90	2	3
41	2	2	91	2	2

42	2	2	92	2	1
43	2	2	93	2	3
44	2	3	94	2	3
45	2	2	95	2	3
46	2	2	96	2	2
47	2	2	97	2	3
48	2	2	98	2	2
49	2	3	99	2	2
50	2	2	100	2	2

表：问题 1 第二组数据

分组序号	元组数量	隐匿数量	分组序号	元组数量	隐匿数量
4	2	1	3	2	51
2	2	2	3	2	52
2	2	3	3	2	53
2	2	4	3	2	54
2	2	5	4	2	55
2	2	6	3	2	56
2	2	7	3	2	57
2	2	8	3	2	58
2	3	9	3	2	59
2	2	10	3	2	60
2	2	11	4	2	61
2	2	12	4	2	62
2	2	13	3	2	63
3	2	14	3	2	64
2	2	15	4	2	65
3	2	16	4	2	66
2	2	17	4	2	67
2	2	18	4	2	68
2	2	19	3	2	69
2	2	20	4	2	70
2	2	21	3	2	71
2	2	22	4	2	72
2	2	23	3	2	73
2	2	24	4	2	74
3	2	25	4	2	75
3	2	26	4	2	76
4	2	27	4	2	77
3	2	28	4	2	78
3	2	29	4	2	79
3	2	30	4	2	80
3	2	31	4	2	81

3	2	32	4	2	82
3	2	33	3	2	83
4	2	34	4	2	84
3	2	35	4	3	85
3	2	36	1	2	86
3	2	37	4	2	87
3	2	38	4	2	88
4	2	39	4	2	89
3	2	40	4	2	90
3	2	41	4	2	91
3	2	42	4	2	92
3	2	43	4	2	93
3	2	44	3	2	94
3	2	45	4	2	95
3	2	46	4	2	96
3	2	47	4	2	97
3	2	48	4	2	98
3	2	49	5	2	99

表：问题 2 第一组数据

...

由于后续数据过多，

隐匿个数	组数	隐匿总数
8	54	
9	94	
10	181	
11	77	
12	22	

隐匿个数	组数
0	39
1	26
2	15

表：问题 P=3 第一组数据

隐匿个数	组数
0	6
1	43
2	69

3	60
4	13
5	9

表：问题 P=3 第二组数据

隐匿个数	组数
2	18
3	49
4	76
5	40
6	4
7	9
8	4

表：问题 P=3 第三组数据

隐匿个数	组数
8	3
9	22
10	71
11	166
12	239
13	253
14	43
15	3

表：问题 P=3 第四组数据

隐匿个数	组数
0	19
1	32
2	29

表：问题 P=5 第一组数据

隐匿个数	组数
1	5
2	56
3	41
4	61
5	32
6	5

表：问题 P=5 第二组数据

隐匿个数	组数
3	25
4	22
5	102
6	51

表：问题 P=5 第三组数据

隐匿个数	组数
10	11
11	80
12	158
13	310
14	158
15	83

表：问题 P=5 第四组数据

隐匿个数	组数
0	32
1	16
2	10
3	22

表：问题 P=8 第一组数据

隐匿个数	组数
2	26
3	55
4	42
5	56
6	11
7	10

表：问题 P=8 第二组数据

隐匿个数	组数
2	10
3	33
4	78
5	42
6	10
7	0

表：问题 P=8 第三组数据

隐匿个数	组数
10	24
11	83
12	117
13	307
14	196
15	46
16	27

表：问题 P=8 第四组数据

单隐藏	全隐藏
59	21

单隐藏	全隐藏
72	128

表： 问题四

单隐藏	全隐藏
1	199
115	
159	

表：问题四

全隐藏
800

表：问题四

1.贪心算法

```
1. import numpy as np
2. from scipy.fftpack import fft
3. from matplotlib.pylab import mpl
4. import pandas as pd
5.
6. import random
7. import copy
8. #数据预处理
9. df = pd.read_excel(r'D:\Desktop\one.xls', sheet_name='二元
   1')
10.data = df.values
11.datatemp = []
12.for i in data:
13.    datatemp.append(i.tolist()[:-2])
```

```
data = datatemp


Datalength = len(data[0])
Listlength = len(data)
templist = []
adrlist = []
Plimit = 2
Group = 1
groupdata = []
tempgrupdata = []
def msglist(list):
    mylist = []
    num = len(list)
    for i in range(Datalength):
        key = 1
        for j in range(num):
            if data[list[j]][i] != data[list[0]][i]:
                key = 0
                mylist.append(0)
                break
        if key == 1:
            mylist.append(1)
    num = 0
    for i in mylist:
        if i == 0:
            num += 1
    mylist.append(num)
    return mylist
def comparedata(list1,list2):
    mylist = []
    unequal = 0
    for i in range(Datalength):
        if list1[i] != list2[i]:
            mylist.append(0)
            unequal = unequal + 1
        else:
            mylist.append(1)
    mylist.append(unequal)
    return mylist

def ifdataequal(list1,list2):
    for i in range(Datalength):
        if list[i] != list2[i]:
```

```
            return 0
    return 1

def datanum():
    num = 0
    for i in data:
        if len(i) == Datalength:
            num += 1
    # print(num)
    return num
def dan():
    mylist = []
    for i in range(len(data)):
        if len(data[i]) == Datalength:
            mylist.append(i)
    return mylist
if __name__ == "__main__":
    for key in range(Datalength + 1):
        print(key)
        # tempgrupdata = []
        for i in range(Listlength - 1):
            print(i)
            if len(data[i]) > Datalength:
                continue
            adrlist = []
            templist = []
            mylist = data[i]
            adrlist.append(i)
            kkke = 0
            for j in range(i + 1, Listlength):
                if len(data[j]) > Datalength:
                    continue
                resultlist = comparedata(mylist, data[j])
                tt = copy.deepcopy(adrlist)
                tt.append(j)
                iftt = msglist(tt)
                if iftt[-1] <= key:
                    templist = resultlist
                    adrlist.append(j)
                if datanum() - len(adrlist) <= Plimit:
                    kkke = 1
                    break
            if kkke == 1:
                adrlist = dan()
```

```
                if len(adrlist) >= Plimit:
                    templist.append(len(adrlist))
                    templist.append(Group)
                    Group = Group + 1
                    for j in adrlist:
                        data[j] = data[j] + templist


                    # tempgrupdata.append(adrlist)
            # groupdata.append(tempgrupdata)

        num = 0
        for i in data:
            num = num + i[-3]
        print(num)


        from xlutils.copy import copy
        import xlrd as xr
        file = "D:\Desktop\T4.2.xls"
        oldwb = xr.open_workbook(file)
        newwb = copy(oldwb)
        newws = newwb.get_sheet(-1 + 1)
        for i in range(len(data)):
            # print(len(data))
            for j in range(len(data[i])):
                # print(data[i])
                newws.write(i + 1, j + 1, data[i][j])  #行，列，数据
        newwb.save(file)
```

2.前三问模拟退火算法

```
import pandas as pd
import random
import math
import numpy as np
import copy


#数据预处理
df = pd.read_excel(r'D:\Desktop\T4.2.xls', sheet_name='Sheet4')
data = df.values
datatemp = []
```

```
for i in data:
    datatemp.append(i.tolist())
data = datatemp
Listlength = len(data)
Datalength = len(data[0])
adrdata = []
rawdata = []
cpadata = []
msgdata = []
rdatalength = int(Datalength/2-2)
groupdata = [0 for i in range(rdatalength+1)]
for i in range(len(groupdata)):
    groupdata[i] = []
for i in data:
    adrdata.append(i[0] - 1)
    rawdata.append(i[1:rdatalength + 1])
    cpadata.append(i[rdatalength + 1:Datalength-3])
    msgdata.append(i[Datalength-3:Datalength])
for i in range(Listlength - 1):
    groupdata[int(msgdata[i][0])].append(i)
listgrp = []
listmsg = []

max = msgdata[0][2]
maxtep = 0
for i in msgdata:
    if i[2] > maxtep:
        maxtep = i[2]

listgrp = [0 for i in range(maxtep)]
listmsg = [0 for i in range(maxtep)]
for i in range(maxtep):
    listgrp[i] = []
    listmsg[i] = []
for i in range(len(msgdata)):
    listgrp[msgdata[i][2] - 1].append(i)
    listmsg[msgdata[i][2]-1] = msgdata[i]

listarea = []
listarea = [0 for i in range(rdatalength + 1)]
for i in range(rdatalength + 1):
    listarea[i] = []
for i in listmsg:
    listarea[i[0]].append(i[2])
```

```
56.
57.listzero = []
58.for i in range(len(listarea)):
59.    if len(listarea[i]) != 0:
60.        listzero.append(i)
61.
62.
63.# print(listgrp)
64.# print(listmsg)
65.# print(rawdata)
66.# print(listarea)
67.# print(listzero)
68.T = 1
69.AT = 0.99
70.
71.
72.
73.templistgrp = copy.deepcopy(listgrp)
74.templistmsg = copy.deepcopy(listmsg)
75.
76.
77.
78.def msglist(list):
79.    mylist = []
80.    num = len(list)
81.    for i in range(rdatalength):
82.        key = 1
83.        for j in range(num):
84.            if rawdata[list[j]][i] != rawdata[list[0]][i]:
85.                key = 0
86.                mylist.append(0)
87.                break
88.        if key == 1:
89.            mylist.append(1)
90.    num = 0
91.    for i in mylist:
92.        if i == 0:
93.            num += 1
94.    mylist.append(num)
95.    return mylist
96.
97.
98.
99.def comparemessagelist(list1,list2):
```

```
    result = []
    for i in range(len(list1)):
        if list1[i] == 0 or list2[i] == 0:
            result.append(0)
        else:
            result.append(1)
    return result
def comparedatalist(list1,list2):
    mylist = []
    unequal = 0
    for i in range(len(list1)):
        if list1[i] != list2[i]:
            mylist.append(0)
            unequal = unequal + 1
        else:
            mylist.append(1)
    mylist.append(unequal)
    return mylist

def comparedata(list1,list2):
    mylist = []
    unequal = 0
    for i in range(rdatalength):
        if list1[i] != list2[i]:
            mylist.append(0)
            unequal = unequal + 1
        else:
            mylist.append(1)
    mylist.append(unequal)
    return mylist

def ifdataequal(list1,list2):
    for i in range(Datalength):
        if list[i] != list2[i]:
            return 0
    return 1
def costfunction(list):
    result = 0
    for i in list:
        result += i[0] * i[1]
    return result

def changeway():
    SUM = 1000
```

```
    while True:
        on = 0
        tw = 0
        tt = 0
        re = []
        fi = []
        se = []
        th = []


        fi.append(listzero[random.randint(0, len(listzero) - 1)])
        fi.append(listarea[fi[0]][random.randint(0, len(listarea[fi[0]]) - 1)])
        fi.append(listgrp[fi[1] - 1][random.randint(0, len(listgrp[fi[1] - 1]) - 1)])

        se.append(listzero[random.randint(0, len(listzero) - 1)])
        while True:
            a = listarea[se[0]][random.randint(0, len(listarea[se[0]]) - 1)]
            if a != fi[1]:
                se.append(a)
                break
            if on > SUM:
                break
            on += 1
        if on > SUM:
            continue
        se.append(listgrp[se[1] - 1][random.randint(0, len(listgrp[se[1] - 1]) - 1)])


        while True:
            b = listzero[random.randint(0, len(listzero) - 1)]
            if (b == fi[0] and b == se[0]) == False:
                th.append(b)
                break
            if tw > SUM:
                break
            tw += 1
```

```
            if tw > SUM:
                continue
            while True:
                c = listarea[th[0]][random.randint(0, len(lis
    tarea[th[0]]) - 1)]
                if c != fi[1] and c != se [1]:
                    th.append(c)
                    break
                if tt > SUM:
                    break
                tt += 1
            if tt > SUM:
                continue
            th.append(listgrp[th[1] - 1][random.randint(0, le
    n(listgrp[th[1] - 1]) - 1)])

            re.append(fi)
            re.append(se)
            re.append(th)
            break
        return re

    def changestate(list):
        #改变listgrp
        global listmsg
        global listgrp
        global listzero
        global listarea
        global templistgrp
        global templistmsg
        templistgrp[list[0][1] - 1].append(list[1][2])
        templistgrp[list[0][1] - 1].remove(list[0][2])

        templistgrp[list[1][1] - 1].append(list[2][2])
        templistgrp[list[1][1] - 1].remove(list[1][2])

        templistgrp[list[2][1] - 1].append(list[0][2])
        templistgrp[list[2][1] - 1].remove(list[2][2])

        #改变msg
        templistmsg[list[0][1] - 1][0] = msglist(templistgrp[
    list[0][1] - 1])[-1]
        templistmsg[list[1][1] - 1][0] = msglist(templistgrp[
    list[1][1] - 1])[-1]
```

```
221.     templistmsg[list[2][1] - 1][0] = msglist(templistgrp[
   list[2][1] - 1])[-1]
222.
223.     return costfunction(templistmsg) - costfunction(listm
   sg)
224. def acceptrule(cost):
225.     if cost < 0:
226.         # print(1)
227.         return 1
228.     else:
229.         one = math.exp((0 - cost) / T)
230.         two = random.randint(0, 100000000) / 100000000
231.
232.         if one >= two:
233.             # print(1,one, two)
234.             return 1
235.         else:
236.             # print(2,one, two)
237.             return 0
238.
239. def acceptstate(ifcost):
240.     # 改变listgrp
241.     global listmsg
242.     global listgrp
243.     global listzero
244.     global listarea
245.     global templistgrp
246.     global templistmsg
247.     global T
248.     # print(T)
249.
250.     if ifcost == 1:
251.         listgrp = []
252.         listmsg = []
253.         listmsg = copy.deepcopy(templistmsg)
254.
255.         listgrp = copy.deepcopy(templistgrp)
256.
257.         listarea = []
258.         listarea = [0 for i in range(rdatalength + 1)]
259.         for i in range(rdatalength + 1):
260.             listarea[i] = []
261.         for i in listmsg:
262.             listarea[i[0]].append(i[2])
```

```
263.
264.             listzero = []
265.             for i in range(len(listarea)):
266.                 if len(listarea[i]) != 0:
267.                     listzero.append(i)
268.
269.             # print(listgrp)
270.             # print(listmsg)
271.             # print(listarea)
272.             # print(listzero)
273.         else:
274.             templistgrp = copy.deepcopy(listgrp)
275.             templistmsg = copy.deepcopy(listmsg)
276.         T = T * AT
277.
278.
279. if __name__ == "__main__":
280.     # a = changeway()
281.     # print(a)
282.     # print(changestate(a))
283.     # print(templistgrp)
284.     # print(listgrp)
285.     # print(templistmsg)
286.     # print(listmsg)
287.     # print(msglist([13,30]))
288.     # print(T < np.power(10.0,-100))
289.     listre = []
290.     listre.append(costfunction(listmsg))
291.     tt = 0
292.     while T > np.power(10.0,-100):
293.         changelist = changeway()
294.
295.         # print(changelist)
296.         cost = changestate(changelist)
297.         acrule = acceptrule(cost)
298.
299.         acceptstate(acrule)
300.         tt = costfunction(listmsg)
301.         listre.append(tt)
302.         print(tt)
303.
304.     print(listgrp)
305.     print(listmsg)
306.
```

```
    from xlutils.copy import copy
    import xlrd as xr
    n = 4
    listans = [[0,listre[0]]]
    key = 0
    for i in listre:
        if listans[key][1] == i:
            listans[key][0] += 1
        else :
            tli = []
            tli.append(1)
            tli.append(i)
            listans.append(tli)
            key += 1
```

```
    file = "D:\\Desktop\\ans.xls"
    oldwb = xr.open_workbook(file)
    newwb = copy(oldwb)
    newws = newwb.get_sheet(-1 + n)
    for i in range(len(listans)):
        for j in range(len(listans[i])):
            newws.write(i + 1, j + 1, listans[i][j])  # 行，列，数据
    newwb.save(file)

    file = "D:\\Desktop\\listmsg.xls"
    oldwb = xr.open_workbook(file)
    newwb = copy(oldwb)
    newws = newwb.get_sheet(-1 + n)
    for i in range(len(listmsg)):
        for j in range(len(listmsg[i])):
            newws.write(i + 1, j + 1, listmsg[i][j])  # 行，列，数据
    newwb.save(file)

    file = "D:\\Desktop\\listgrp.xls"
    oldwb = xr.open_workbook(file)
    newwb = copy(oldwb)
    newws = newwb.get_sheet(-1 + n)
    for i in range(len(listgrp)):
```

```
            for j in range(len(listgrp[i])):
                newws.write(i + 1, j + 1, listgrp[i][j])  # 行，列，数据
        newwb.save(file)
```